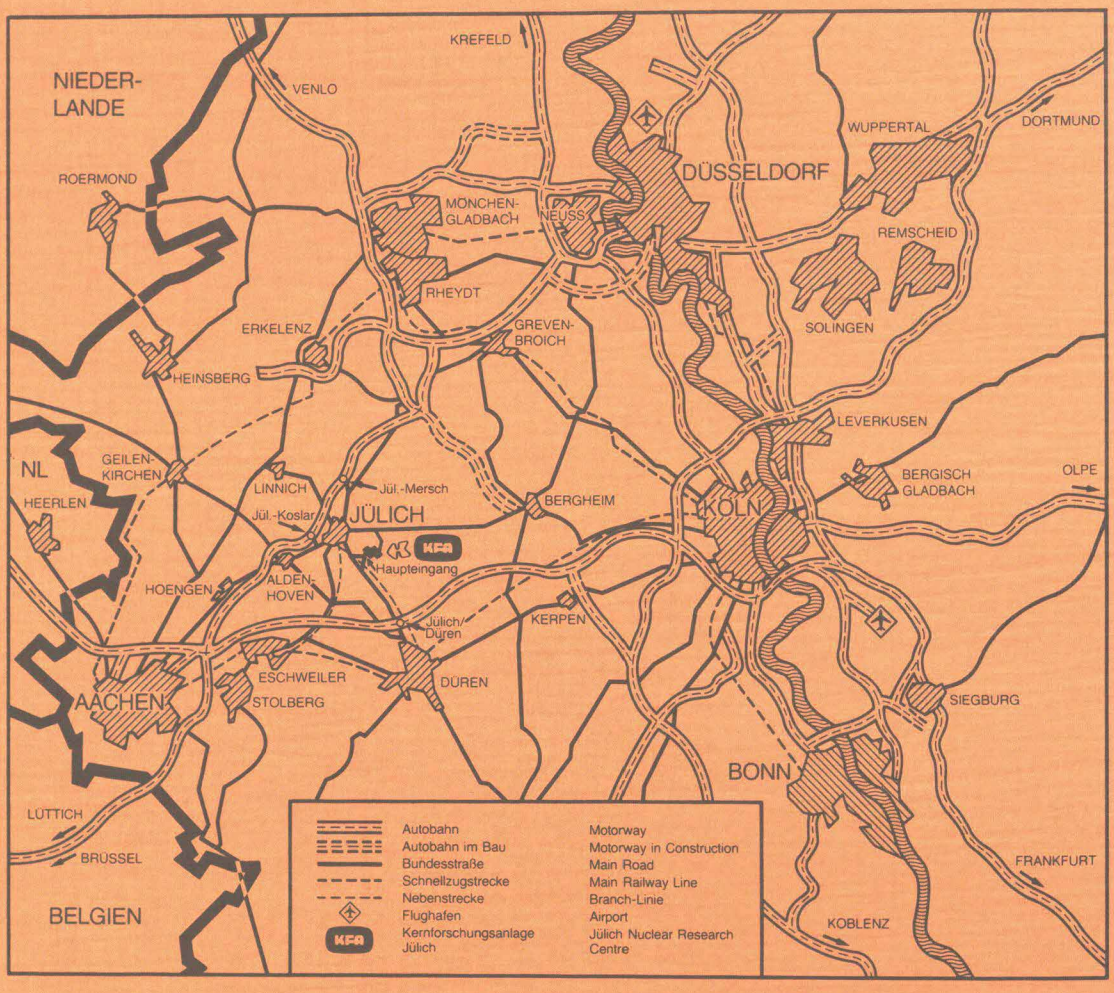
KERNFORSCHUNGSANLAGE JÜLICH GmbH
Arbeitsgruppe für Schichten- und Ionentechnik

Prozeßautomatisierung einer dreiachsigen Magnetfeldmeßmaschine

von

U. Böhm

Jül - Spez - 353
März 1986
ISSN 0343-7639



Als Manuskript gedruckt

Spezielle Berichte der Kernforschungsanlage Jülich – Nr. 353

Arbeitsgruppe für Schichten- und Ionentechnik Jül - Spez - 353

Zu beziehen durch: ZENTRALBIBLIOTHEK der Kernforschungsanlage Jülich GmbH
Postfach 1913 · D-5170 Jülich (Bundesrepublik Deutschland)
Telefon: 02461/610 · Telex: 833556-0 kf d

Prozeßautomatisierung einer dreiachsigen Magnetfeldmeßmaschine

von

U. Böhm

INHALTSVERZEICHNIS

VORWORT

In der Kernforschungsanlage Jülich ist ein Ringbeschleuniger geplant. Um den Teilchenstrahl auf bestimmten Bahnen zu halten, werden Dipolmagnete (als Biegemagnete) und Quadrupole (als Fokussiermagnete) benötigt. Zur Realisierung der Ionenoptik werden in der Strahlführung und im Ring die verschiedenen Magnettypen eingesetzt.

Bei Quadrupolmagneten und Dipolmagneten muß der aus den optischen Anforderungen resultierende Feldverlauf mit engen Toleranzen (besser als 10^{-4} mm) eingehalten werden.

Die Feldeigenschaften werden mit Hilfe von Hallproben gemessen. Bei den verwendeten Hallproben ist der Ort der Feldmessung mechanisch sehr genau bekannt.

Um mit den Hallproben an einen definierten Meßpunkt anzufahren, wird ein dreiachsiger Meßtisch benötigt. Dieser Meßtisch soll die Hallproben mit einer Genauigkeit von 10^{-2} mm an diesen Punkt für die Feldmessung fahren.

Um ein schwingungsfreies Anfahren und Abbremsen zu ermöglichen braucht man eine Prozeßsteuerung. Diese Prozeßsteuerung beinhaltet auch ein automatisches Vermessen der Magnetfelder.

1 EINLEITUNG

Für die Vermessung von Magnetfeldern wird ein Meßgerät benötigt, das ein automatisches Vermessen des Feldes, dreidimensional ermöglicht. Die Automatisierung wird durch eine prozeßgesteuerte Feldmeßmaschine ermöglicht.

Diese Feldmeßmaschine besteht aus einem dreiachsigen Koordinatenmeßtisch, auf dem ein Hallprobenträger angebracht ist (s. Bild 1). Die Achsen des Koordinatenmeßtisches werden durch Gleichstromscheibenläufermotoren angetrieben, da diese im Gegensatz zu Steppermotoren, ein schwingungsarmes Anfahren und Abbremsen ermöglichen. Das Anfahren erfolgt über ein lineares Ansteigen der Motorspannung (Rampe), bis zu einem vorgegebenen Spannungswert. Das Abbremsen erfolgt mit der gleichen Prozedur allerdings mit einer negativen Steigung.

Da sich beim Anfahren der Positionen mit Hilfe von Gleichstromscheibenläufermotoren die anzufahrende Position nicht durch Zählen der Umdrehungen feststellen läßt, werden an den Achsen Lichtmaßstäbe angebracht. Diese Lichtmaßstäbe besitzen außer der Information der Positionen einen Referenzimpuls, aus dem mechanisch die absolute Position bezogen wird, sowie Rechts-Linksinformationen.

Die Signalaufbereitung für den Meßtisch erfolgt in gesonderten Steuereinschüben. Für die Gleichstromscheibenläufermotoren wird eine Motoransteuerung benötigt, die Signalspannungen in die entsprechenden Leistungssignale für den Motor umsetzt. Die Lichtmaßstäbe benötigen eine Signalvorverarbeitung und -verstärkung der Lichtimpulse von den Achsen des Koordinatenmeßtisches. Dabei wird die Rechts-Linksinformation ausgewertet.

Als Interface zwischen Rechner und Steuereinschüben kommt ein intelligentes Kartensystem der Firma Hewlett Packard vom Typ 6942A, Multiprogrammer, zum Einsatz. Die Verbindung zwischen Prozeßrechner und Interface erfolgt über den Industriestandard IEEE-488-Bus. Der Prozeßrechner HP1000-A600 der Firma Hewlett Packard hat eine speziell für eine schnelle Datenübertragung ausgelegte IEEE-488-Busansteuerung.

Die Interfacekarten werden benötigt, um die digitalen Werte vom Rechner in analoge Signale für die Motoransteuerung umzuwandeln und die digitalen Impulse der Lichtmaßstäbe zu zählen, sowie die Referenzimpulse auszuwerten. Der Multiprogrammer empfängt Werte über den Zustand der Endschalter an den Achsen des Meßtisches. Diese Signale werden als TTL-Signale (Low-Level: 0-0.5 V, High Level: 2.0-5.0 V) über die Steuereinschübe an den Multiprogrammer weitergegeben. Außerdem steht ein NOT-AUS zur Verfügung, um eventuelle Beschädigungen zu vermeiden.

Der Multiprogrammer wird vom Rechner aus durch entsprechende Programme bedient. Als Benutzerschnittstelle für diese Software wird ein interaktiver Monitor eingesetzt, der eine einfache Bedienung der Prozeßsteuerung ermöglicht. Dieser Monitor startet interaktiv Routinen, die die Informationen der Feldmeßmaschine verarbeiten und synchronisieren. Der Monitor beinhaltet auch einen direkten Zugriff auf die Daten der Feldmeßmaschine ("ONLINE CONTROL").

2 BESCHREIBUNG DER GERÄTE

Es werden benutzt: 1 Koordinatenmeßtisch
3 Steuereinschübe
1 Positioniercontroller
(local-remote control)
1 Multiprogrammer
2 Rechner

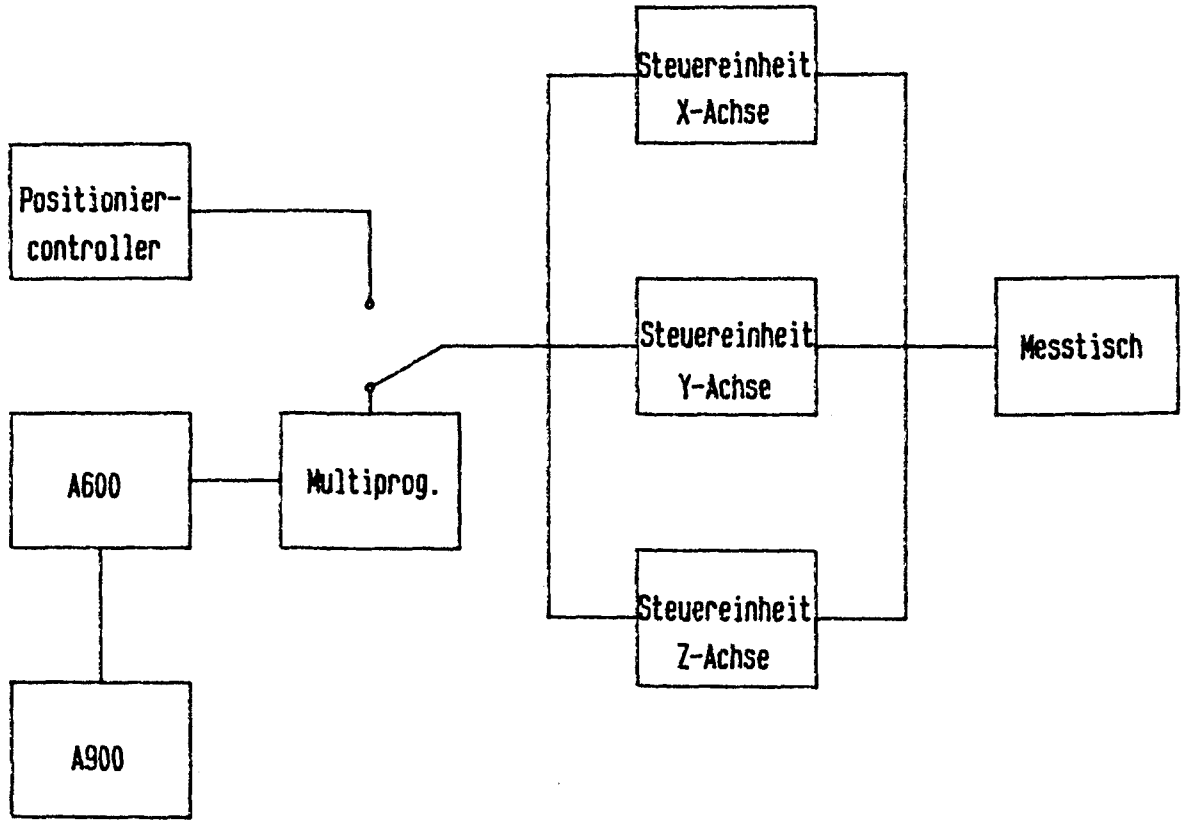


Abb. 1 Anordnung der Geräte

2.1 Meßtisch

Der Meßtisch besteht aus einem Betonsockel, einer 150 mm starken Granitplatte und drei aufmontierten, beweglichen Achsen (X-,Y-,Z-Achse).

Der Betonsockel soll die Schwingungen, die vom Boden auf den Meßtisch übertragen werden können, aufnehmen. Darüber befindet sich eine Granitplatte, die eine sehr gute Ebenheit aufweist ($5*10^{-6}$ m). Zwischen Betonblock und Granitplatte befinden sich drei Justierschrauben und zwei Stützschrauben. Mit Hilfe der drei Justierschrauben kann die Granitplatte in Höhe und Neigung justiert werden, so daß sie sich waagerecht ausrichten läßt. Durch die beiden Stützschrauben wird ein Kippen der Platte verhindert.

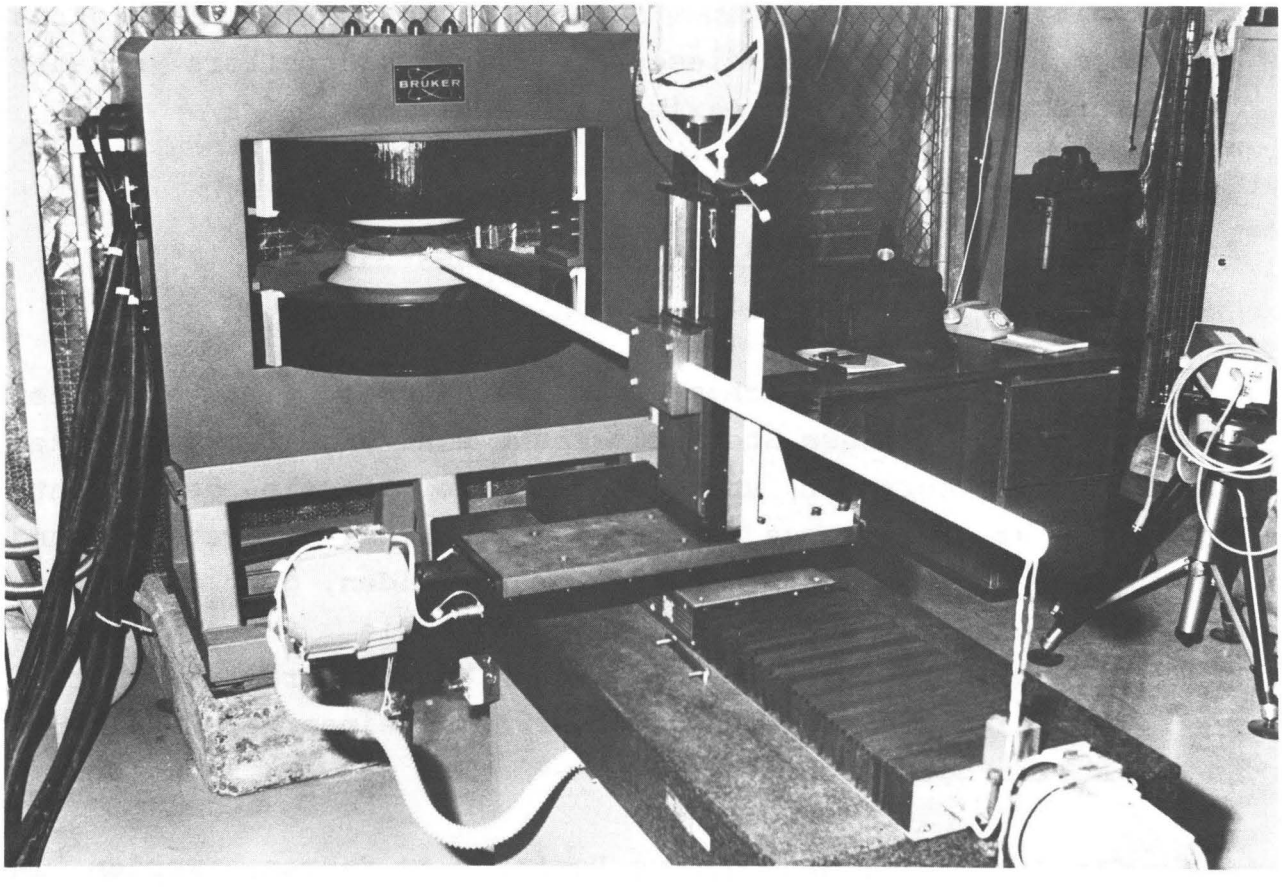


Bild 1: Meßtisch mit Achsen, Hallprobenträger und Magnet

Auf der Granitplatte ist die X-Achse fest montiert. Der Schlitten der X-Achse trägt die Y-Achse, auf der wiederum die Z-Achse montiert ist. Der Hallprobenträger ist an der Z-Achse befestigt. Die drei Achsen stehen senkrecht zueinander. An jeder Achse befindet sich ein Lichtmaßstab, der aus zwei im Abstand von $20*10^{-6}$ m übereinanderliegenden Strichgittern aufgebaut ist, wobei ein Strichgitter gegenüber dem anderen $10*10^{-6}$ m versetzt ist. In der Mitte jedes Lichtmaßstabes befindet sich ein Referenzimpuls. /1/

2.2 Positioniercontroller

Der Positioniercontroller ermöglicht dem Benutzer einzeln programmierbare Positionen auf den Achsen des Meßtisches anzufahren. Der Speicher dieses Positioniercontrollers erlaubt die Vorprogrammierung von maximal 100 Positionen. /2/

2.3 Steuereinschübe als Signalvorverarbeitung

Für jede der drei Achsen des Meßtisches gibt es einen Steuereinschub. Diese Steuereinschübe sind Signalvorverstärker für den Multiprogrammer und wandeln die Signale der Lichtmaßstäbe des Meßtisches durch Signaltreiber (Verstärker und Entkoppler) um. Um Beschädigungen zu vermeiden, gibt es erstens einen NOT-AUS und zweitens geben die Steuereinschübe Endschaltersignale vom Meßtisch an den Multiprogrammer weiter. Die Steuereinschübe geben von jeder Achse einen Referenzimpuls an den Multiprogrammer weiter.

Um die Achsen auf eine genaue Position zu fahren, werden in den Steuereinschüben das 0 Grad und das um 90 Grad phasen-

verschobene Signal umgewandelt (siehe Impulsdiagramm). Wenn eine Achse nach rechts gefahren wird, stehen +90 Grad an und wenn die Achse nach links gefahren wird, stehen -90 Grad an. Diese Signale werden in einem UND-Gatter verbunden und gehen anschließend in zwei Flip-Flops, die entweder durch die negative oder positive Flanke getriggert werden. Dabei sperrt je ein Flip-Flop das andere. Daraus folgt eine Rechts-Linksinformation. Das 0 Grad Signal setzt ein Fenster, um einen definierten Ausgangszustand im Flip-Flop zu erzeugen. Weiter wird durch das 0 Grad Signal über ein Monoflop ein Bereitschaftssignal erzeugt, das eine Interfacekarte des Multiprogrammers in einen empfangsbereiten Zustand versetzt. Durch den Referenzimpuls wird die Interfacekarte des Multiprogrammers auf einen vordefinierten Wert gesetzt. /3/

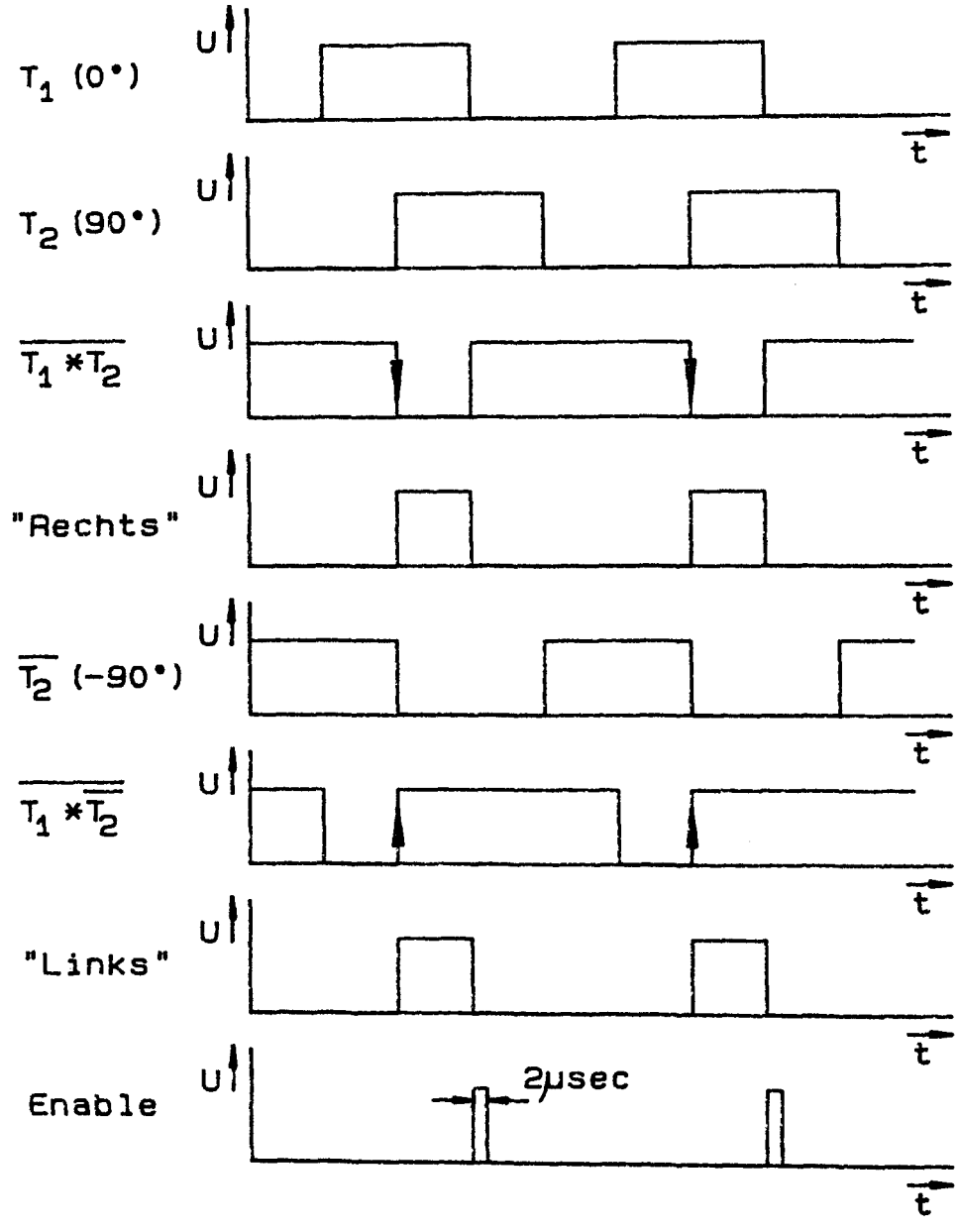


Abb. 2 : Impulsdiagramm für Rechts- Linksinformation für je eine Achse

2.4 Multiprogrammer (HP Modell 6942A)

Der Multiprogrammer ist ein Interfacesystem zur Prozeßautomatisierung. Es dürfen in Verbindung mit einem Extender HP Modell 6943A, bis zu acht Multiprogrammereinheiten zusammengeschlossen werden. In jedem Multiprogrammer können bis zu 16 Interfacekarten eingesteckt werden, also insgesamt 128 Interfacekarten. /4/

Die CPU des Multiprogrammers akzeptiert 800 000 Bytes pro Sekunde und kann 128 Character mit einer Rate von 23 000 Character pro Sekunde im Buffer speichern. Der Multiprogrammer stellt folgende Funktionen zur Verfügung:

OP (Output Parallel): Damit können mehrere Interfacekarten so programmiert werden, daß alle Karten zur gleichen Zeit Werte ausgeben.

IP (Input Parallel): Wartet, bis alle Interfacekarten ihre Daten empfangen haben. Erst dann werden die Daten vom Multiprogrammer gelesen und abgespeichert.

WF (Write First Rank): Schreibt Werte in einen Speicher 1.Ordnung der Interfacekarte. Die Funktion WF ist ein Befehl, der nur im Zusammenhang mit anderen Befehlen (z.B. II,OP,...) benutzt werden kann.

II (Input Interrupt): Wartet, bis eine Interfacekarte einen Prozeß abgeschlossen hat, speichert dann die Daten der Karte und ihre Adressen im Buffer des Multiprogrammers ab. Anschließend setzt der Multiprogrammer

ein Signal an den Rechner ab (Service Request).

RV (Read Value): Liest Daten einer (oder mehrerer) Interfacekarte(n) und speichert diese im Multiprogrammerspeicher.
RV initiiert keinen Lesezyklus, daher stammen die gespeicherten Werte immer vom letzten, abgeschlossenen, Lesezyklus.

AC (Arm Card): Versetzt eine Interfacekarte in einen Wartezustand, so daß sie von einem externen Triggersignal angesprochen werden kann.

SF (Set Format): Diese Funktion erlaubt ein Ändern der Datenformate der Interfacekarten.

2.4.1 Syntax für WRITE-Befehle /5/

Folgende Syntax wird für einen WRITE-Befehl benutzt:

OPCODE,A1,D1,A2,D2,...,T

OPCODE: ist eine der in Kapitel 2.4 beschriebenen Funktionen

A1,A2,...: sind Slotnummern des Multiprogrammers für die 1.Karte,2.Karte,usw.

D1,D2,...: sind Daten für die Karten

T: ist der Terminator

Diese Syntax gilt für jede in dieser Arbeit verwendeten Funktion, mit Ausnahme der Funktion SF (Set Format). Hier sieht die Syntax folgendermaßen aus:

SF,A1,#P,[DATA TYPE,][LSB,][SIZE,][LIM,][ID,]A2,#P,...,T

SF: Set Format

A1,A2...: Slotnummern

#P: Anzahl der Parameter: 0 bis 4,-1,-2,-5
0 = Default (Standardannahme)
1 bis 4 = Format einer Funktionskarte
-1 = Zahl der Interrupts innerhalb

100 Millisekunden
-2 = Wechselstromfrequenz (50 bis 60 Hz)
-5 = Format einer Karte ohne Funktion
(nur für Tests)

DATA TYPE: Code für den Datentyp

LSB: Wert für LSB (Low Significant Bit)

SIZE: 12 oder 16 Bit

LIMIT: kleinster programmierbarer Wert

ID: Kennzahl einer Karte

T: Terminator

Als Beispiel die im Programm verwendete Syntax für die SF-Funktion :

SF,Slot#,2,0.005,T

Slot#: Slotnummer für DAC-Karte (s. Kapitel 2.5.1)

2: Data Type = Dezimal (Sign/Magnitude Binary)

0.005: LSB

T: Terminator

2.4.2 Syntax für READ-Befehle /5/

Die Daten aus dem Multiprogrammer werden alle von einer Extended Talk Address (s. Kapitel 2.7) gelesen.

Syntax:

LU:XX,Slot#,T

LU: Logical Unit = HP-IB Adresse

XX: Extended Talk Address: 1 = IP
2 = II
6 = RV
10 = Multiprogrammerstatus
11 = Fehlerliste
12 = Armed Card Interruptlist

Slot#: Slotnummer der Interfacekarte

T: Terminator

2.5 Interfacekarten für Multiprogrammer /4/

Die Eingänge und Ausgänge der Interfacekarten sind durch Optokoppler oder Impulsüberträger galvanisch voneinander getrennt um Beschädigungen durch falsch angelegte Spannungen an der Eingangsseite der Karten zu vermeiden. In den folgenden

Kapiteln werden die bei der Prozeßautomatisierung verwendeten Interfacekarten beschrieben.

2.5.1 Digital to Analog Voltage Converter Card (HP Modell 69720A)

Diese Karte (DAC) besitzt einen 12 Bit Digital/ Analog-Wandler, womit eine Spannung in einem Bereich von -10.24 V bis +10.235 V mit einem minimalen Schritt von 0.005 V erzeugt werden kann. Spannungen, die außerhalb des angegebenen Bereiches liegen, werden nicht ausgegeben. Es erfolgt aber eine Fehlermeldung durch die CPU des Multiprogrammers.

Werden Formate verwendet, die nicht von der Firma eingestellt wurden, können diese durch eine SF-Anweisung oder durch Umlegen von Schaltern auf der Karte geändert werden. Eine vorher eingestellte Ausgangsspannung kann durch eine Programminstruktion oder durch ein externes Triggersignal,z.B. von einer Timerkarte, abgesetzt werden.

2.5.2 Timer/Pacer Card (HP Modell 69736A)

Diese Timer/Pacer Karte (T/P) kann Rechteckimpulse mit einer Impulsdauer von einer Mikrosekunde bis 2^{16}-1 (65535) Sekunden erzeugen. Sie kann entweder in einem one-shot mode (nur jeweils ein Impuls) oder in einem recirculate mode (Wiederhol-Mode) betrieben werden. Die Anfangszeit der Signale kann durch einen Programmbefehl oder durch ein externes Triggersignal bestimmt werden.

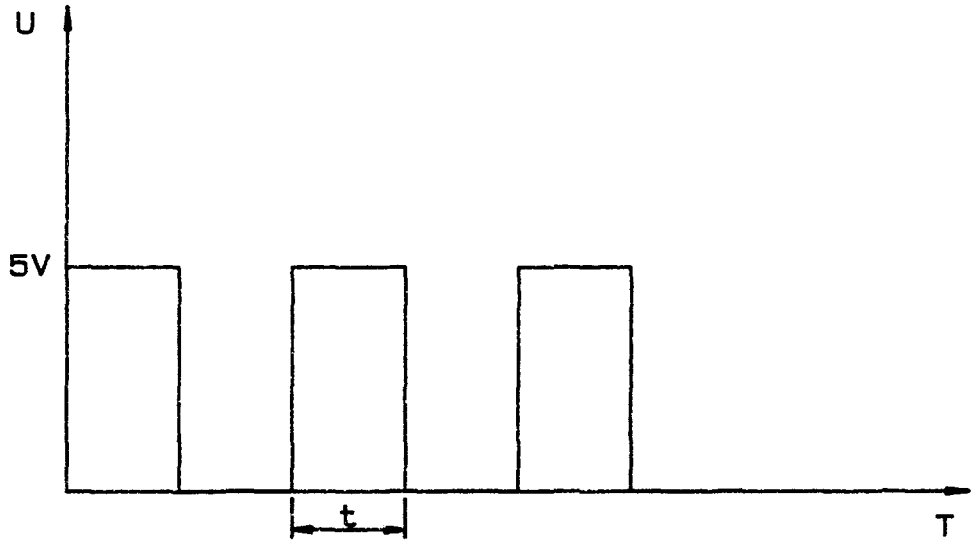


Abb. 3: Impulsdiagramm der Timer/Pacer Karte

Das Format dieser Karte kann auf zwei Wegen bestimmt werden:

1. Durch Gebrauch einer SF-Instruktion.

2. Durch Verwendung der Sekundäradresse 2 (s. Kapitel 2.6), in Verbindung mit einer WF-Instruktion.

Im zweiten Fall kann der recirculate-mode folgendermaßen gesetzt werden:

0 = Mikrosekunden

1 = Millisekunden

2 = Sekunden

Nachfolgend ein Beispiel für den beschriebenen Befehl:

WF,7.2,1T,OP,7,10T

WF,OP: OPCODES

7: Slotnummer in der die Karte steckt

2: Sekundäradresse 2

1: Millisekunden

10: Pulsdauer von 10 Millisekunden

T: Terminator

2.5.3 Isolated Digital Input Card (HP Modell 69770A)

Mit dieser Karte (DIGIN) wird ein 16 Bit Wort gelesen. Die Datenübertragung kann wieder durch einen Programmbefehl oder durch ein externes Triggersignal ausgelöst werden. Der Multiprogrammer liest dann das Wort vom Speicher der Karte und schreibt es in seinen Speicher. Es können auf diese Weise insgesamt 1339 Wörter im Multiprogrammer abgespeichert werden. Dieses 16 Bit Wort ergibt ein Integerwort mit einem Wertebereich von 0 bis maximal $2^{16}-1$ (65535). Ein Signal mit einer Spannung zwischen -6.1 und 1.0 Volt wird als logische Null und Signal zwischen 4.0 und 6.1 Volt als logische Eins verstanden.

Es besteht auch die Möglichkeit das 16 Bit Wort an eine Wortinterruptkarte weiterzugeben, wodurch Interrupts erzeugt werden können (s. Kapitel 2.5.5).

2.5.4 Counter/Totalizer Card (HP Modell 69775A)

Die Counterkarte ist in der Lage von Null an aufwärts bis $2^{16}-1$ oder von $2^{16}-1$ abwärts bis auf Null mit einer Frequenz von einem Megahertz zu zählen. Wenn mehrere Counterkarten (max. 128) zu einer Kaskade zusammengeschlossen werden, kann der Zählbereich bis auf $2.56*10^{18}$ erhöht werden. Dabei wird die Zählfrequenz auf 500 KHz limitiert. Der Startpunkt zum Zählen wird durch ein Enablesignal festgesetzt. Eine Counterkarte kann in folgenden drei Schritten programmiert werden (s. auch Kapitel 2.6):

1. Referenzcount zwischen 0 und 2^{16}-1 ins Presetregister laden.

2. Zirkulieren der Karte durch Programm oder externes Triggersignal.

3. Lesen der gezählten Pulse.

2.5.5 Interrupt Card (HP Modell 69776A)

Die Interruptkarte vergleicht ein 16 Bit Referenzwort mit einem Inputwort und erzeugt einen Interrupt, wenn eine bestimmte Bedingung erfüllt wird. Das Inputwort kann ungleich, gleich, größer oder kleiner als das Referenzwort sein. Es können auch ein oder mehrere Bits miteinander verglichen werden, wenn eine Maske in der Karte programmiert wurde. Mit diesem Interrupt kann z.B. eine Programmprozedur abgebrochen werden.

Zum Beschreiben der Karte, können drei Sekundäradressen verwendet werden (s. Kapitel 2.6):

1. Sekundäradresse 0: Auf dieser Adresse kann ein Referenzwort in die Karte geschrieben werden. Das Referenzwort ist ein Integerwort zwischen 0 und 2^{16}-1.

2. Sekundäradresse 1: Mit Hilfe dieser Adresse wird ein Interruptmode in Oktalwerten programmiert. Der Interruptmode vergleicht ein Inputwort mit dem Referenzwort. Der Mode besteht aus Octalwörtern. Diese sind:

1 bedeutet ungleich
2 bedeutet gleich
4 bedeutet größer als
10 bedeutet kleiner als

3. Sekundäradresse 2: Auf dieser Adresse wird eine Maske in die Karte programmiert. Diese Maske vergleicht, ob sich einzelne Bits verändern.

Zum Lesen werden folgende Sekundäradressen benutzt:

0 --- des Wortes, das den Interrupt auslöst

1 --- des Inputwortes

2 --- der Maske

3 --- des Referenzwortes.

2.6 Sekundäradressen für Interfacekarten /5/

Wie schon erwähnt gibt es verschiedene Sekundäradressen für WRITE und READ Befehle. Diese Sekundäradressen werden an die Slotnummer angehängt. Slotnummern und Sekundäradressen werden durch einen Punkt getrennt. Wird keine Sekundäradresse angegeben, gilt der Standardwert Null.

Für die verwendeten Karten haben die Sekundäradressen folgende Funktionen:

	WRITE Subaddresses				READ Subaddresses			
Cardtyp	0	1	2	3	0	1	2	3
DAC	Output Value	—	—	—	—	—	—	Output Value
Timer	Pulse Width	Period Multipl.	Mode	—	—	—	—	Pulse Width
Iso.Dig. Input	Self-test Value	—	—	—	Input Value	—	—	Self-test Value
Counter	Count Preset	—	—	—	Present Count	—	—	Count Preset
Word Interrupt	Refer. Word	Mode	Mask	—	Interrupt Word	Extern. Word	Mask	Refer. Word

Tab. 1: Card Subaddresses

2.7 Verdrahtung des Multiprogrammers

Die Verbindung zwischen den einzelnen Interfacekarten und dem Prozeß erfolgt über Postverteiler. Hier können die einzelnen Signale rangiert werden und bieten den Vorteil, über Meßadapter am Rangierverteiler Funktionstests bzw. Fehlersuche durch

Signalanalyse, zu betreiben.

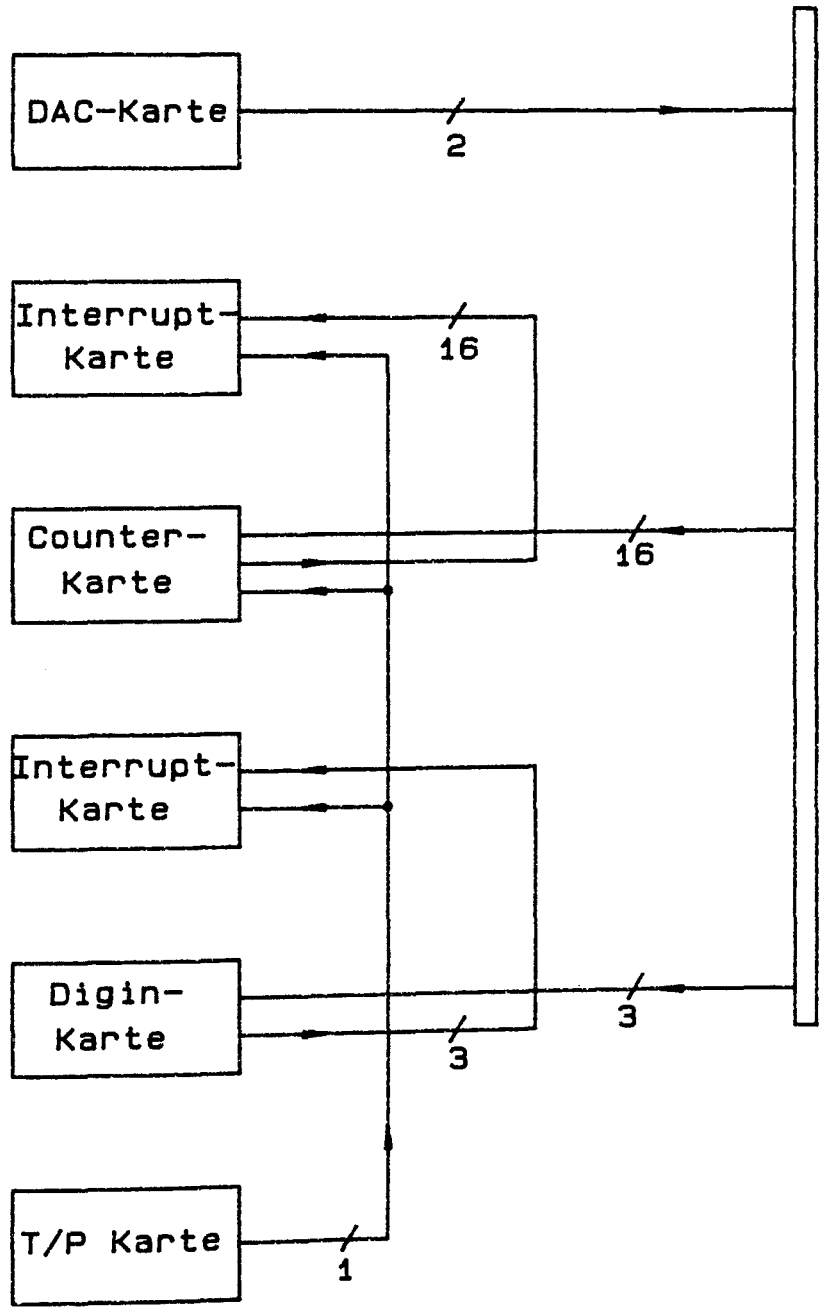


Abb. 4: Verdrahtungsschema des Multiprogrammers für eine Achse

2.8 HP-IB-Bus

Der verwendete HP-IB-Bus, entwickelt von der Firma Hewlett Packard, entspricht der IEC-Norm. Der allgemeine Name eines solchen Busses ist GPIB (General Purpose Interface Bus). Die internationale Normung lief parallel beim IEEE (Institute of Electrical and Electronics Engineers) und in der IEC (International Electrotechnical Commision). Die Normen sind in den Normblättern IEC 625-1 und 625-2, DIN IEC 625 und IEEE-488/1978 festgelegt.

Der Bus kann als Controller, als Sprecher (Talker) und als Hörer (Listener) verwendet werden. An den HP-IB-Bus können bis zu 15 Geräte angeschlossen werden. Dabei entspricht ein Gerät etwa einem Meter Buskabel, d.h. der Anschluß eines Gerätes schwächt die Signale in gleichem Maße, wie ein 1 m langes Kabel. Erstreckt sich die zu überbrückende Entfernung über mehr als 15 Meter, müssen die Signale durch Extender verstärkt werden. Die Gesamtlänge der Busleitung beträgt 15 m. Die Übertragungsgeschwindigkeit bei Maximallast ist 20 Kbyte/s und bei 0,5 m Kabellänge pro Gerät bis zu 1 Mbyte/s. Der Signalpegel ist ein TTL (Transistor-Transistor-Logic) mit Negativlogik (False als logische 0 = 2.5...5 V und True als logische 1 = 0...1.4 V).

Gesendet werden Zeichen aus dem ASCII-Zeichensatz. Der Bus selbst verhält sich passiv, d.h. alle Aktivitäten gehen vom Rechner oder von den angeschlossenen Steuergeräten aus.

Um die Verbindung zwischen Rechner und dem angeschlossenen Gerät herzustellen, werden Geräteadressen (Logical Units) verwendet. Da die meisten Geräte noch Subadressen besitzen, gibt es die Möglichkeit, an die Geräteadressen noch Sekundäradressen anzuhängen. Diese werden von der Geräteadresse durch einen Doppelpunkt getrennt (s.auch Kapitel 2.4.2).

Um auf spontan auftretende Ereignisse reagieren zu können, ist eine Alarmverarbeitung (Interrupt Handling) notwendig (Echtzeitverarbeitung). Die dafür benötigte Unterbrechungssteuerung muß den Zentralprozessor (CPU) veranlassen, nach einer Meldung eines Steuergerätes (Alarm- oder Interruptanforderung, bzw. Service Request,SRQ) die gerade laufenden Aktionen definiert zu unterbrechen, das Steuergerät zu bedienen (Interrupt-Service) und exakt danach in das unterbrochene Programm zurückzukehren. Ein Service Request ist ein Bedienungaufsruf eines Steuergerätes um z.B. einen Interrupt auszulösen. /6/ /7/

2.9 Rechner

Verwendet wird ein Mikrorechner der Firma Hewlett Packard. Der Rechner ist eine HP1000-A600. Die A600 ist ein 16 bit Mikrocomputer mit einer Geschwindigkeit von ca. 1 Mips (1 Mio. Instruktionen pro Sekunde) mit einem Hauptspeicher von 1.2 MByte, einer Winchesterplatte von 67 MByte und einem 65 MByte Streamertape. Außerdem gibt es einen 8 Kanal Multiplexer für RS 232 C Schnittstellen, Terminal- und Druckeranschluß. Für den Datenbus werden vier IEEE-488 Interfacekarten verwendet.

Weiterhin gibt es noch zwei Punkt-Punkt Rechnerkopplungen. Eine zur HP1000-A700 und eine zur HP1000-A900. Die A900 wird nur zur Softwareentwicklung benutzt, wogegen die A600 als reiner Prozeßrechner verwendet wird.

3 BESCHREIBUNG DER PROGRAMME

Die Programme werden in Standard-FORTRAN 77 geschrieben, da es auf den Rechnern der HP1000'er-Serie ein ausgesprochen benutzerfreundliches Betriebssystem für Fortran 77 gibt. Das Fortran 77 basiert auf dem ANSI 77 Standard mit Erweiterungen entsprechend dem MIL-STD-1753 Standard (Military Standard Fortran). /6/

Das Programm für die Prozeßsteuerung besteht aus einem Monitor und unterlagerten Routinen, die als selbständige Programme ausgeführt sind, für die Ansteuerung des Meßtisches. Der einfache Befehlsaufbau des erstellten Monitors macht die Handhabung der Feldmeßmaschine für den Benutzer sehr einfach. Da alle Programme die gleichen Variablen benutzen, wird ein gemeinsamer Datenbereich benutzt, ein sogenannter shared Commonblock. Auf diesen Datenbereich können mehrere eigenständige Programme zugreifen. Die Synchronisation der einzelnen Programme erfolgt über Steuersignale, den Resourcenummern (RSN).

3.1 Programmierter Zugriff auf das Betriebssystem

Unter dem RTE-A Betriebssystem stehen für Programme zwei Arten der Kommunikation mit dem Betriebssystem zur Verfügung.

Der Zugriff auf den Kernspeicher erfolgt durch EXEC-Befehle. Das sind insbesondere alle Aufrufe für die Programmablaufsteuerung wie z.B. Start und Stop, warten auf Ereignisse usw. und alle direkten I/O Operationen, die unmittelbar auf die entsprechenden Interfacekarten zugreifen. Hier ist für das Folgende die Installation von

Interruptserviceroutinen von besonderer Bedeutung.

Zum Anderen besteht die Möglichkeit des programmierten Zugriffs auf das Filesystem der HP1000. Diese Aufrufe beinhalten auch das Laden von Programmen von Platte und das Laden, kombiniert mit dem Starten von Programmen. Dieser Zugriff verbirgt sich hinter den FMP-Aufrufen.

Als Sonderfunktion stehen auch einige Aufrufe zur Verwaltung von Betriebsmitteln zur Verfügung. Für das vorliegende Problem ist allerdings nur die Verwaltung der Resourcenummern durch die Funktion RNRQ von Bedeutung. /9/

3.2 Programmaufbau

Im Folgenden wird das Zusammenwirken der einzelnen Programme erläutert. Durch den Monitor TAC werden die Befehle und Parameter eingelesen. Der Monitor untersucht die eingegebenen Zeichenketten mit Hilfe der MD-Routinen und durch ein Computed GOTO wird in die einzelnen Unterprogramme verzweigt (s. Flußdiagramm 1).

Als erstes Unterprogramm muß TAC_INIT aufgerufen werden (s. Flußdiagramm 6). Dieses springt nach der Initialisierung der Startwerte für die Achsen und für den Multiprogrammer in das Unterprogramm TAC_INIT_SUB. Durch die Funktion

```
IERROR = FMPRPPROGRAM (CPNAME,RPNAME,OPTION,IERROR)
```

IERROR Integervariable (Rückgabeparameter).
Ist sie Null, dann gab es keinen Fehler,

ist sie < 0, dann wurde ein Fehler entdeckt.

CPNAME Programmname (TACSR)

RPNAME Gibt den Programmnamen zurück.

OPTION Eingabeparameter der Optionen definiert (hier ohne Bedeutung).

wird das Programm TACSR geladen. Das Programm TACSR wird dann als Interrupt-Serviceroutine mit Hilfe des Aufrufs

```
CALL SRQ (ILU_P,IVALUE,IPNAME)
```

ILU_P Geräteadresse

IVALUE Integerwert, der an IPNAME übergeben wird

IPNAME Name des Serviceprogramms

installiert und aktiviert. TAC_INIT_SUB läuft währenddessen weiter und holt durch

```
CALL RNRQ (22B,IRN,ISTAT)
```

22B reserviert die RSN

IRN gibt die allocated RSN zurück

zwei Resourcenummern aus dem Betriebssystem des Rechners. Die

Resourcenummern sollen die Programme synchronisieren und die laufenden Programme gegen äußere Eingriffe schützen. Der Oktalwert 22B bewirkt, daß die Resourcenummern für den Prozeß allocated und gelockt werden. Die zwei Resourcenummern werden für die Programme TACRI und TACPR gebraucht. Das TACSR wird durch einen Interrupt vom Multiprogrammer gestartet und läuft bis zu dem Befehl

```
CALL EXEC (6,0,1)
```

6 stoppt das Programm

0 für das rufende Programm

1 versetzt das Programm in einen Wartezustand und sichert die Daten.

Das Programm bleibt im Rechner, behält die letzten Daten und ist für einen erneuten Aufruf durch einen weiteren Interrupt bereit. Damit ist das Unterprogramm TAC_INIT_SUB beendet. TAC_INIT ruft anschließend durch die Funktion

```
IERROR = FMPRUNPROGRAM ('TACRI',IPRAMS,RUNNAME)
```

IERROR wie bei FMPRPPROGRAM

'TACRI' Programmname

IPRAMS gibt Programmparameter zurück

RUNNAME gibt Programmname zurück

das Programm TACRI auf, um eine Referenzimpulsfahrt zu starten. TAC_INIT wartet bis das Programm TACRI beendet ist.

TACRI initialisiert zuerst die Interfacekarten des Multiprogrammers und startet die Referenzimpulsfahrt. Danach wird die Interruptkarte aktiviert und TACRI wartet dann durch

```
CALL RNRQ (ICNTWAIT,IRN,ISTAT)
```

ICNTWAIT = 2B locked RSN

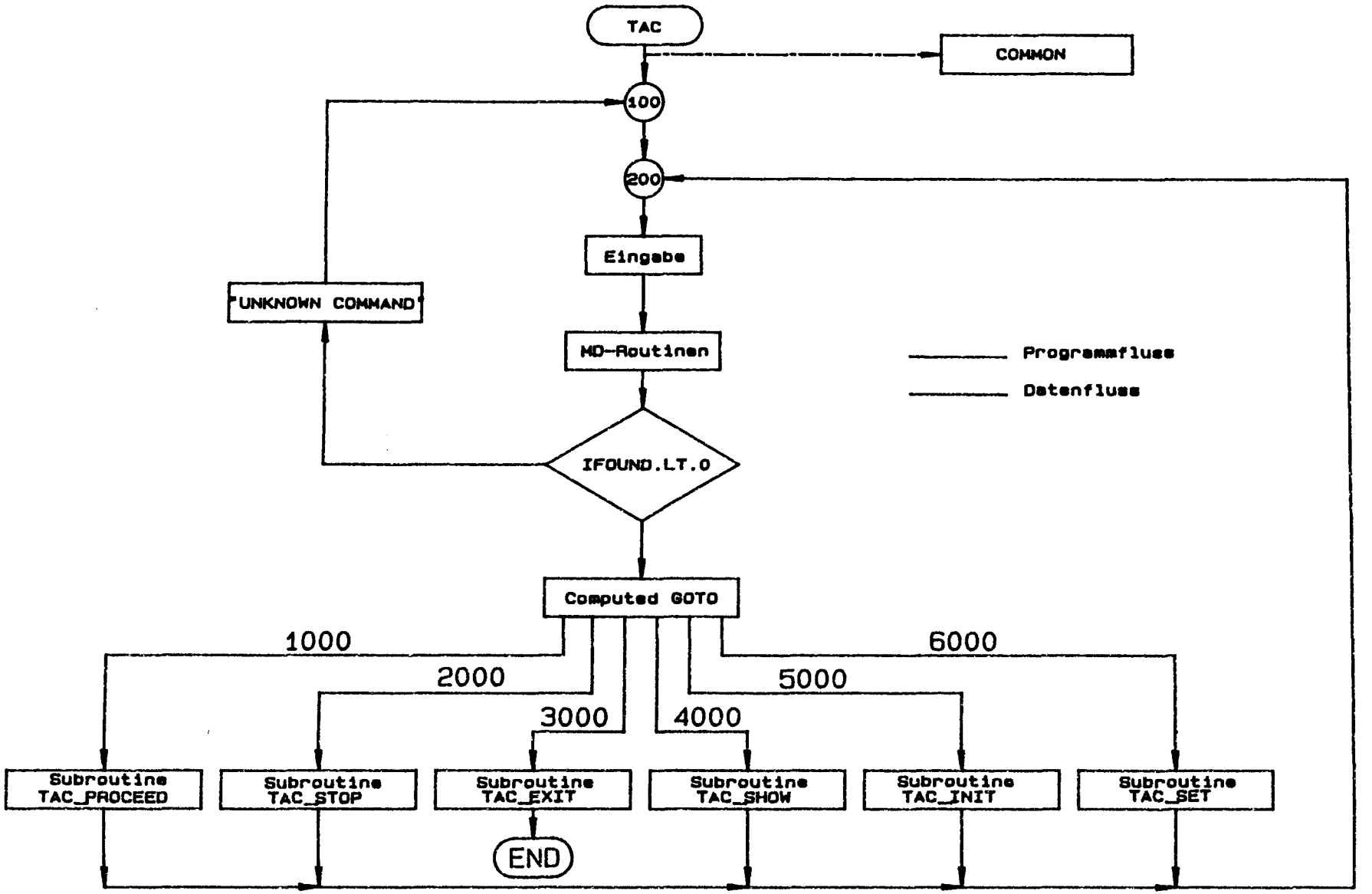
IRN und ISTAT wie schon beschrieben (s. Seite 26)

auf einen Interrupt, der durch die Endschalter aktiviert wird. Wenn ein Interrupt ansteht, wird durch den Multiprogrammer ein Service Request (SRQ) mit der entsprechenden Resourcenummer an den Rechner abgegeben. Dieser startet dann das Programm TACSR, das den Interrupt auf der Interruptkarte löscht und die Resourcenummer zurücksetzt. Danach springt das Programm wieder auf den EXEC 6-Befehl und wartet auf einen neuen Interrupt. TACRI dagegen ändert das Vorzeichen der Fahrtrichtung der gesetzten Achse, damit der entgegengesetzte Endschalter angefahren und somit der Referenzimpuls aufgenomme werden kann. Anschließend wird die Interruptkarte erneut aktiviert und die Resourcenummer gesetzt. Wenn die Achse auf den nächsten Endschalter fährt, setzt der Multiprogrammer erneut ein SRQ ab und TACSR durchläuft nochmals die beschriebene Prozedur. TACRI liest dann die Position der Achse und schreibt den aktuellen Wert in den Commonblock. Das Programm TACRI und das Unterprogramm TAC_INIT sind damit beendet.

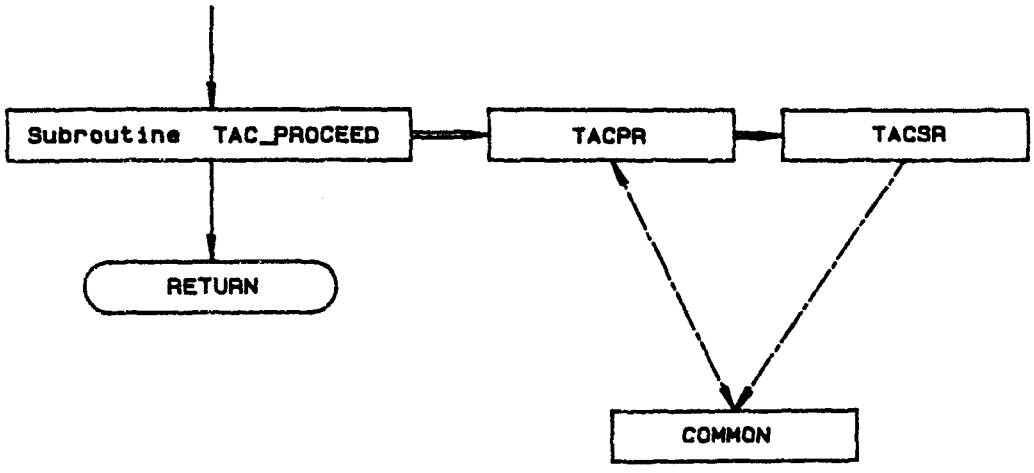
Als nächstes können die Positionen und die Geschwindigkeiten der Achsen gesetzt werden (s. Flußdiagramm 2).

Danach kann dann TAC_PROCEED zur Positionsfahrt der Achsen aufgerufen werden. Von TAC_PROCEED wird das Programm TACPR durch einen FMPRUNPROGRAM-Befehl (s. Seite 27) gestartet. Dort werden die Positionen und die Geschwindigkeiten aus dem Commonblock gelesen und die Positionsfahrt gestartet. In einem bestimmten Abstand (der von der Geschwindigkeit abhängig ist) vor der gewünschten Position, wird ein Interrupt abgesetzt und ein SRQ mit der zweiten Resourcenummer an den Rechner abgegeben (s. TACRI), der dann wieder das Programm TACSR startet. Das Programm TACPR verringert die Geschwindigkeit allmählich, bis die gewünschte Position erreicht ist. Hier ist die Geschwindigkeit gleich Null. Die aktuelle Position der Achse wird gelesen und in den Commonblock geschrieben. Damit ist das Programm TACPR und das Unterprogramm TAC_PROCEED beendet.

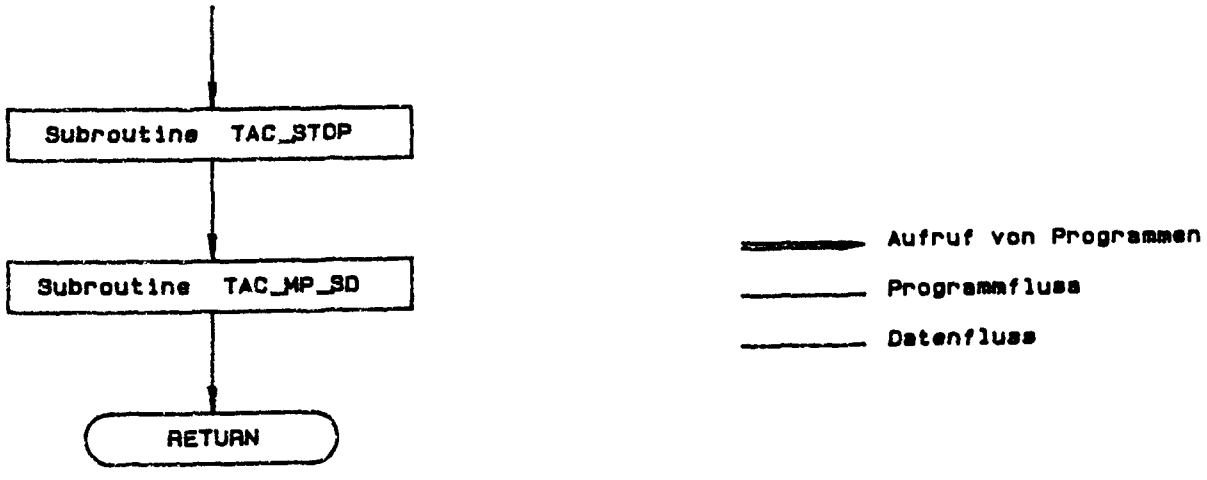
Soll der Monitor gestoppt werden, wird durch das Unterprogramm TAC_EXIT die Routine TAC_EXIT_SUB aufgerufen. Dort werden die in TAC_INIT_SUB reservierten Resourcenummern wieder an das Betriebssystem zurückgegeben (s. Flußdiagramm 4).



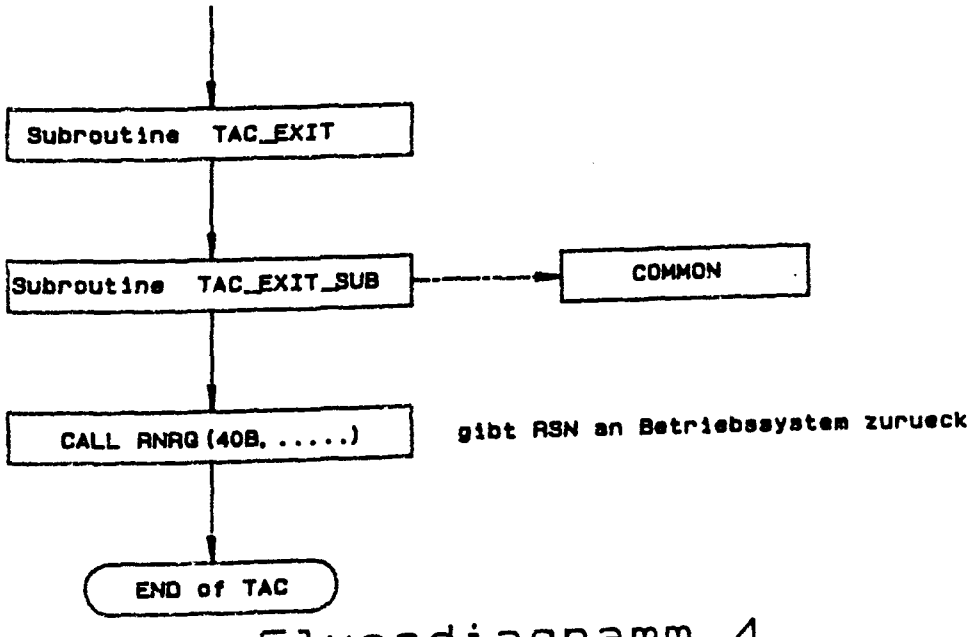
Flussdiagramm 1



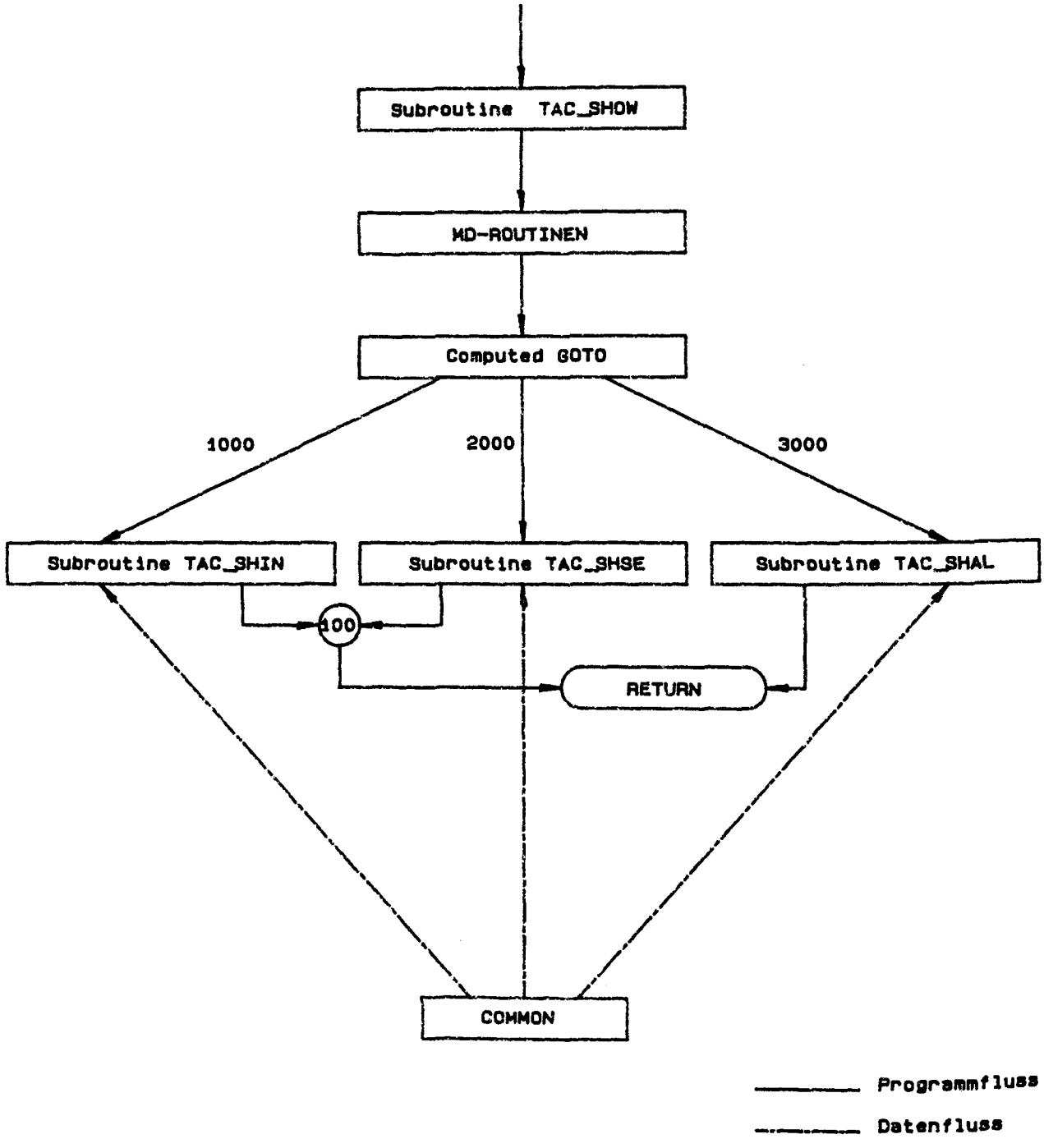
Flussdiagramm 2



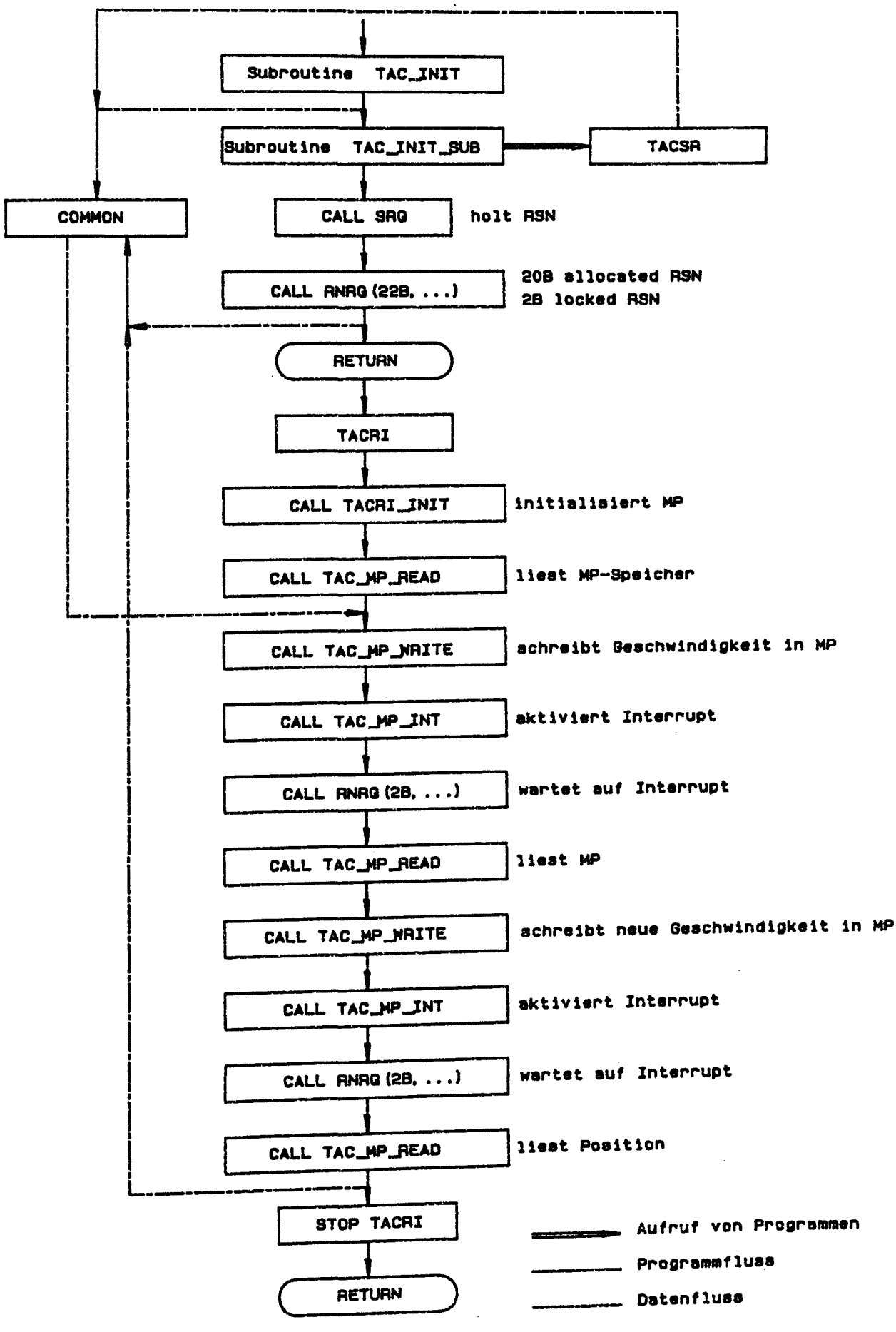
Flussdiagramm 3



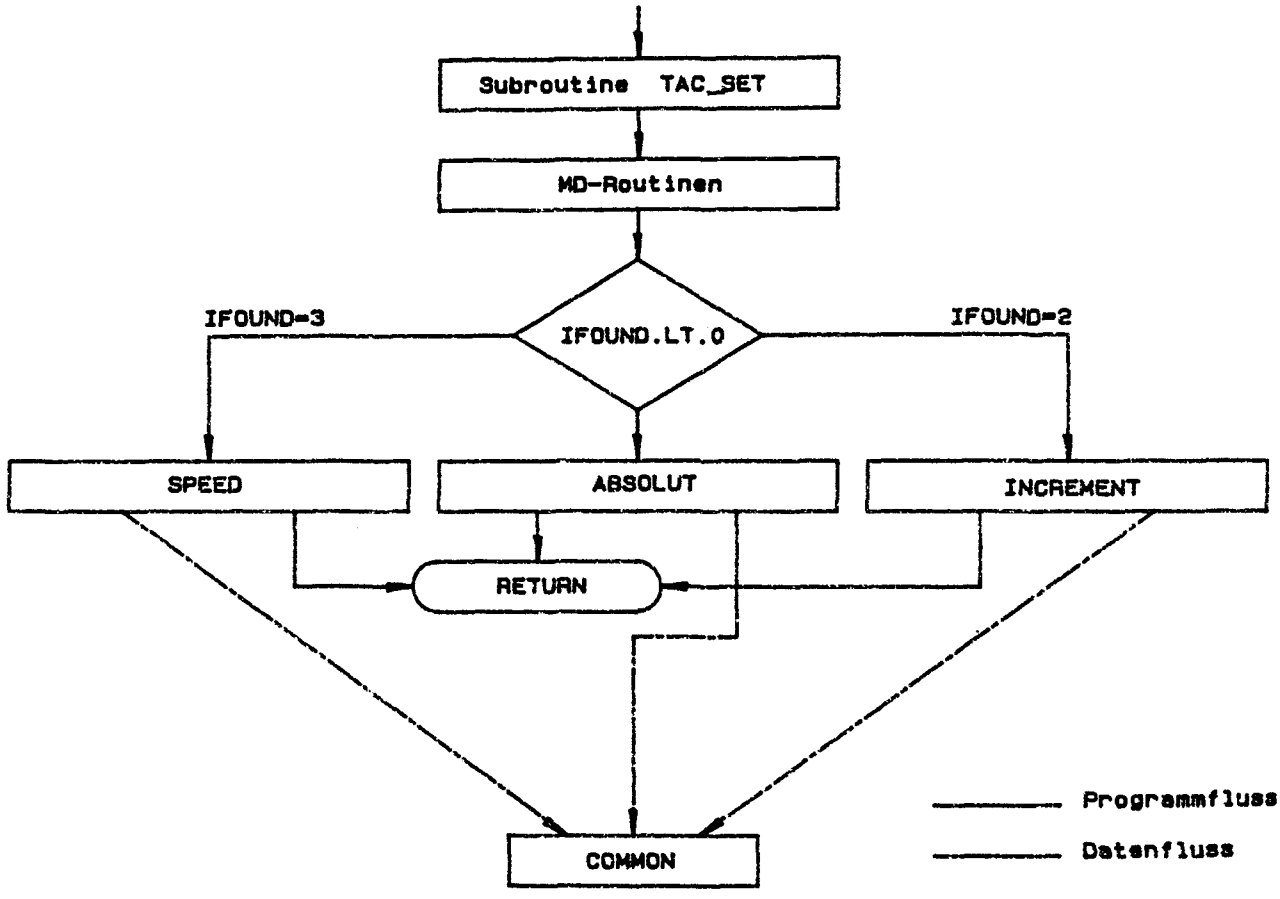
Flussdiagramm 4



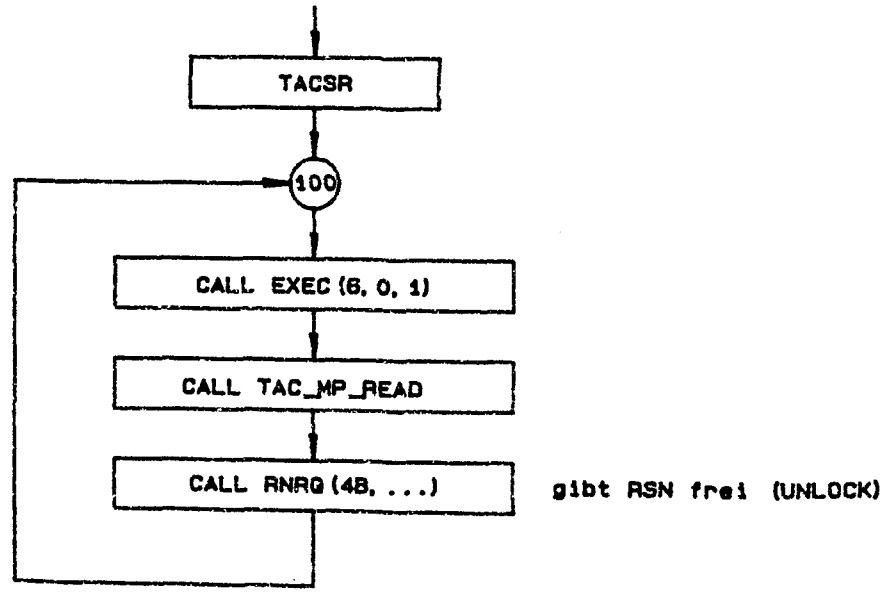
Flussdiagramm 5



Flussdiagramm 6



Flussdiagramm 7



Flussdiagramm 8

3.3 Monitor TAC

Durch den Monitor werden Benutzeranweisungen interpretiert und die gewonnenen Parameter im gemeinsamen Datenbereich gespeichert. Der Monitor aktiviert die unterlagerten Programme, die dann ihrerseits die Multiprogrammerbefehlsstrukturen generieren und an den Multiprogrammer weitergegeben. Gestartet wird der Monitor mit dem Befehl TAC (Three Axis Controller). /10/

3.4 Unterprogramm TAC_INIT

In dem Unterprogramm TAC_INIT werden zunächst die Positionswerte für die Achsen im Commonblock auf Null gesetzt und die Geschwindigkeit der Achsen auf einen mittleren Wert. Weiterhin werden die Geräteadresse des Multiprogrammers und die Slotnummern für die Multiprogrammerinterfacekarten in den Commonblock geschrieben. Diese Werte werden alle aus einem Konfigurationsfile gelesen.

Anschließend wird das Unterprogramm TAC_INIT_SUB aufgerufen. Dieses Unterprogramm lädt dann das Programm TACSR. TACSR läuft dann bis zu einem EXEC-Befehl und wartet auf einen Interrupt. TAC_INIT_SUB dagegen reserviert für die Synchronisation der einzelnen Programme zwei Resourcenummern und schreibt diese in den Commonblock. Anschließend erfolgt der Rücksprung von TAC_INIT_SUB nach TAC_INIT.

Danach wird vom Unterprogramm TAC_INIT das Programm TACRI (s. Kapitel 3.4) für eine Referenzimpulsfahrt gestartet. TAC_INIT springt nach Beendigung der Referenzimpulsfahrt ins

Hauptprogramm zurück.

3.5 Programm TACRI

Das Programm TACRI startet eine Referenzimpulsfahrt und erlaubt dem Multiprogrammer einen Interrupt abzusetzen. Nachdem der Interrupt aktiviert wurde wartet das Programm darauf, daß die Resourcenummer zurückgesetzt wird. Durch erreichen eines Endschalters wird ein Interrupt ausgelöst. Der Multiprogrammer gibt sofort nachdem er von der Interruptkarte den Interrupt empfangen hat, einen Service Request (SRQ) an den Rechner ab. Dieser SRQ startet das Programm TACSR. TACSR läßt dann das Programm TACRI weiterlaufen. Damit wird das Vorzeichen der Geschwindigkeit geändert und die Achse auf den anderen Endschalter gefahren. Nach zwei Sekunden wird ein neuer Interrupt aktiviert und das Programm wartet erneut auf einen Interrupt. Dieser wird dann ausgelöst, wenn die Achse auf den nächsten Endschalter fährt. Nachdem dies geschehen ist, wird erneut ein Interrupt ausgelöst und ein SRQ abgesetzt. Damit ist das Programm beendet und der Monitor TAC meldet sich zurück.

Zwischen den zwei Interrupts wird ein Referenzimpuls aufgenommen, der bewirkt, daß die Counterkarte einen beim INIT festgesetzten Wert erhält.

Die Unterprogramme TAC_MP_WRITE und TAC_MP_READ bewirken, daß einerseits Befehlsstrukturen an den Multiprogrammer übermittelt werden und andererseits Informationen, z.B. Statusinformationen, ausgelesen werden können. Die Interruptkarten werden durch das Unterprogramm TAC_MP_INT in einen für Interrupts empfangsbereiten Zustand versetzt.

3.6 Unterprogramm TAC_SET

Im Unterprogramm TAC_SET werden die Positionen und Geschwindigkeiten der Achsen gesetzt. Dies geschieht durch die Verwendung von MD-Routinen, die es ermöglichen, Befehlsstrukturen zu interpretieren. Die gesetzten Werte werden dann in den Commonblock geschrieben (s. Flußdiagramm 7).

3.7 Unterprogramm TAC_PROCEED

Das Unterprogramm TAC_PROCEED startet durch eine FMP-Funktion das Programm TACPR für eine Positionsfahrt. TAC_PROCEED wartet dann solange bis TACPR beendet ist und kehrt dann zum Monitor TAC zurück.

3.8 Programm TACPR

Durch das Unterprogramm TAC_PROCEED wird das Programm TACPR aufgerufen, das eine Positionsfahrt startet. Es wird zunächst eine Rampe mit linearer Steigung, zur Beschleunigung der Achse, gefahren /1/. Diese Rampe verhindert ein dynamisches Schwingen der Achse und somit auch des Hallprobenträgers. TACPR wartet dann auf einen Interrupt.

Wenn die Achse, die durch SET definierte Position erreicht hat, wird der Interrupt ausgelöst. Dieser Interrupt startet das Programm TACSR, welches dann das Programm TACPR startet. Das Programm TACSR bremst die Achse mit einer negativen Steigung der Rampe ab. Durch die sehr schnelle Verarbeitung des Interruptes wird die Achse mit einer Toleranz von ± 0.02 mm an die

gewünschte Position gefahren. Diese Toleranz entspricht dem Abstand zweier Lichtpunkte auf den Lichtmaßstäben der Achsen.

3.9 Unterprogramm TAC_STOP

Das Unterprogramm TAC_STOP ruft die Subroutine TAC_MP_SD auf. Diese Subroutine versetzt den Multiprogrammer in einen Zustand, der alle Werte der Outputkarten auf Null setzt und diese im Multiprogrammerspeicher absichert. Somit können die Achsen sofort gestoppt werden (s. Flußdiagramm 3).

3.10 Unterprogramm TAC_SHOW

Das Unterprogramm TAC_SHOW, TAC_SHIN und TAC_SHSE lesen aus dem Commonblock die aktuellen Positionen und Geschwindigkeiten. Desweiteren werden auch die Geräteadresse des Multiprogrammers und die Slotnummern der Interfacekarten ausgelesen und auf das Terminal ausgegeben. Die Subroutine TAC_SHOW wird dabei wieder durch die Verwendung der MD-Routinen und dem Computed GOTO in die Unterprogramme TAC_SHIN und TAC_SHSE verzweigt (s. Flußdiagramm 5).

3.11 Unterprogramm TAC_EXIT

Durch TAC_EXIT wird das Unterprogramm TAC_EXIT_SUB aufgerufen. Dieses Unterprogramm gibt die in TAC_INIT_SUB reservierten Resourcenummern an das Betriebssystem zurück. Danach beendet TAC_EXIT den Monitor TAC.

3.12 Shared Commonblock

Der gemeinsame Datenbereich wird für alle Programme durch den Commonblock TACBL bestimmt. In dem Commonblock stehen alle Werte, die im Programm TAC und seinen interaktiven Routinen benötigt werden. Jedes Programm hat also Zugriff auf diesen Commonblock, d.h. jedes Programm liest und schreibt Werte in diesen Commonblock. Um die Werte kontrollieren zu können, kann der Commonblock durch das Programm TACBR (Tac-Block-Read) gelesen werden. Das ist besonders zur Fehlersuche in der Kommunikation zwischen den Programmen wichtig. Die Werte des Commonblockes werden durch TACBR auf das Terminal gelistet.

3.13 Befehlsaufbau

Die Handhabung der Befehle und deren Parameter soll für den Benutzer möglichst einfach sein. Diese Anforderung wurde erfüllt durch:

1. Mehrere Befehle und Parameter können in einer Zeile eingegeben werden. Sie werden durch Leerzeichen getrennt.

2. Für nicht gegebene Parameter wird eine Standardannahme gemacht.

3. Die Befehle dürfen auf die ersten zwei Zeichen gekürzt werden. Die Befehle EXIT und STOP dürfen bis auf das erste Zeichen gekürzt werden.

4. Die Parameter können als Integer, Real, mit und ohne Exponent eingegeben werden.

Bei der Eingabe ist es wichtig, daß die Befehle und ihre Parameter verschiedene Namen haben. Dadurch wird erkannt, welche Parameter vorhanden sind, und für welche Standardannahmen gesetzt werden müssen. Die nachfolgenden Befehle können somit richtig erkannt und abgearbeitet werden.

3.14 Funktionsweise des Monitors

Der Aufruf des Monitors erfolgt durch TAC. Ist der Monitor gestartet, so meldet er sich mit dem Prompt

TAC>

Danach kann die Eingabe von Befehlen und deren Parameter erfolgen. Zwischen den Befehlen und ihren Parametern muß immer ein Leerzeichen stehen.

Zur Untersuchung der Befehle werden MD-Routinen benutzt. Dies sind Routinen, die die Befehle und Parameter interpretieren. Die einzelnen Befehle werden in separaten Unterprogrammen bearbeitet. In die Unterprogramme wird mit Hilfe einer Fallunterscheidung verzweigt, ausgeführt als Computed GOTO. Die damit gewählte Struktur ermöglicht eine einfache Analyse von Befehlen und läßt die Möglichkeit offen, zu einem späteren Zeitpunkt sehr einfach Erweiterungen anzubringen.

3.15 Befehle zur Ausführung von TAC

Als Befehle zur Steuerung des Koordinatenmeßtisches sind vorgesehen:

PROCEED	Startet eine Positionsfahrt
STOP	Stoppt alle Funktionen (vergleichbar mit NOT-AUS)
EXIT	Beendet TAC
INIT	Initialisiert den Multiprogrammer und startet die Referenzimpulsfahrt (definierter Anfangszustand)
SET	Setzt Position und Geschwindigkeit der Achsen
SHOW	Zeigt die Daten des Commonblockes

In einer 72 Zeichen langen Zeile können beliebig viele Kommandos eingegeben werden. Der genaue Aufbau der Befehle die Bedeutung der Parameter ist im Folgenden dargestellt.

Parameter, die fehlen dürfen, sind in eckige Klammern gesetzt. Alternative Angaben sind von geschweiften Klammern eingeschlossen und durch ein Oder-Zeichen (|) getrennt.

3.15.1 INIT

Der Aufruf INIT darf bis auf die ersten beiden Zeichen gekürzt werden. Beim INIT werden die Slotnummern für die Interfacekarten des Multiprogrammers, die Gerätenummer, die maximale und minimale Position und der größtmögliche Geschwindigkeitsbereich aus einem Konfigurationsfile gelesen und in den für die Programme gemeinsamen Datenbereich geschrieben.

Durch INIT werden auch die drei Achsen auf Null positioniert und die Geschwindigkeiten auf einen mittleren Wert gesetzt.

Der Aufruf INIT erfolgt durch

```
TAC>IN[IT]
```

3.15.2 SET

Mit SET wird für eine Achse die Position und die Geschwindigkeit eingegeben. Bei jeder Eingabe wird kontrolliert, ob die eingegebenen Werte innerhalb der vereinbarten Grenzen liegen. Wenn die Positionsangabe außerhalb des Fahrbereiches liegt, erhält der Benutzer auf dem Terminal die Meldung, daß der eingegebene Wert falsch ist und es wird um eine neue Eingabe gebeten.

Bei einer zu großen oder zu kleinen Geschwindigkeit wird der Wert entweder auf +10 Volt oder -10 Volt festgelegt. Auf dem Terminal erscheint dann der neue Wert. Bei einer richtigen

Eingabe erfolgt keine Meldung.

Bei der Positionsangabe werden die Befehle ABSOLUT,INCREMENT und SPEED benutzt. Alle drei Befehle dürfen bis auf das erste Zeichen gekürzt werden. Bei Eingabe des Befehles INCREMENT mit anschließendem Wert setzt sich die neue Position, aus der alten Position, plus (bzw. minus) dem neuen Wert zusammen.

Die Geschwindigkeit wird durch den Befehl SPEED plus dem Wert der Geschwindigkeit in Volt angegeben.

Allgemeiner Befehlsaufbau von SET:

TAC>SE[T] {x|y|z} {A[BSOLUT]|I[NCREMENT]|S[PEED]} [VALUE]

Nachfolgend ein Beispiel für eine Befehlszeile, mit der die X-Achse auf die Position -251.59 mm mit einer Geschwindigkeit von 2.450 Volt gefahren wird:

TAC>SE X ABS -251.59 SE X SP 2.450

3.15.3 PROCEED

Mit PROCEED wird das Anfahren der mit SET gesetzten Position eingeleitet. Der Monitor wartet bis die anzufahrende Position erreicht ist.

Der Aufruf lautet:

TAC>PR[OCEED]

3.15.4 STOP

Durch den Aufruf

TAC>S[TOP]

wird die Fahrprozedur sofort gestoppt. STOP übernimmt also eine ähnliche Funktion wie ein NOT-AUS.

3.15.5 SHOW

Durch den Aufruf

TAC>SH[OW] {SE[T]|IN[IT]|[AL[L]]}

werden die Werte des Commonblocks ausgegeben. Zwischen SH[OW] und einem der Parameter SE[T], IN[IT] und [AL[L]] steht genau ein Leerzeichen. Die Parameter dürfen bis auf die beiden ersten Zeichen gekürzt werden.

Durch den Aufruf

```
TAC>SH[OW] SE[T]
```

werden die Positionen und die Geschwindigkeiten der drei Achsen gezeigt.

Durch

```
TAC>SH[OW] IN[IT]
```

werden die Slotnummern für die Achsen und die Geräteadresse für den Multiprogrammer gezeigt.

Durch

```
TAC>SH[OW] [AL[L]]
```

zeigt SET und INIT zusammen auf dem Bildschirm (Standardannahme).

3.15.6 EXIT

Durch den Aufruf

```
TAC>E[XIT]
```

werden die Resourcenummern wieder freigegeben und das Programm

```
TAC beendet.
```

4 SCHLUßWORT

In der vorliegenden Arbeit wurden die gestellten Anforderungen für eine Prozeßautomatisierung einer dreiachsigen Magnetfeldmeßmaschine durch die entwickelten Programme erfüllt. Alle Programme wurden in FORTRAN 77, basierend auf dem ANSI 77 Standard mit Erweiterungen entsprechend dem MIL-STD 1753 Standard. Die Programme sind durch die MD-Routinen und einem Computed GOTO so aufgebaut, daß

1. der Benutzer eine einfache Bedienung der Steuerung vorfindet und

2. eventuelle Erweiterungen der Programme einfach nur angehängt werden.

Für die Vereinfachung der Programme wurden für die Variablen der einzelnen Programme, gleiche Namen benutzt. Dadurch kann für alle Programme ein gemeinsamer Datenbereich benutzt werden.

Die Synchronisation der Programme erfolgt über Resourcenummern, die vom Betriebssystem geholt werden.

Soll die Toleranz für die anzufahrenden Positionen verkleinert werden, können anstatt der Lichtmaßstäbe die Achsen mit Laser vermessen werden. Die Laservermessung, in Verbindung mit dem vorhandenen Programm, würde ein noch genaueres Positionieren der Achsen ermöglichen.

5 LITERATURVERZEICHNIS

/1/ Vermessung von dynamischen Eigenschaften eines dreiachsigen Koordinatenmeßtisches, Ingenieurarbeit von R. Schetzke, FH Aachen, Abt. Jülich, Mai 1985

/2/ Bedienungshandbuch Positioniercontroller GEL 4400 von Lenord + Bauer & Co GmbH, März 1984

/3/ N. Bongers, Kreuztischdokumentation, unveröffentlicht

/4/ Using the 9826 and 9836 Computers with the 6942A Multiprogrammer, HP Part Number 06942-90013, August 1982

/5/ 6942A Multiprogrammer Quick Reference Guide, HP Part Number 06942-90004, July 1980

/6/ H. Schumny, Digitale Schnittstellen in Computersystemen und Übertragungsnetzen, Vorlesung im Haus der Technik e.V. Essen, Oktober 1984

/7/ The HP-IB in HP1000 Computersystems Users Manual, HP Manual, Part Number 59310-90064, December 1983

/8/ HP1000 L-Serie Computer I/O Interfacing Guide, HP Manual Part Number 02103-90005, April 1982

/9/ RTE-A User's Manual, HP Manual Part Number 92077-90002, January 1985

/10/ IGM Interaktiver Graphischer Monitor, Ingenieurarbeit von U. Hacker, FH Aachen, Abt. Jülich, September 1979

/11/ Regionales Rechenzentrum für Niedersachsen, Universität Hannover, FORTRAN 77 Sprachumfang, Revision B/Januar 1984

/12/ FORTRAN 77 Reference Manual, HP Manual Part Number 92836-90001, February 1985

/13/ RTE-A Programmer's Reference Manual, HP Manual Part Number 92077-90007, January 1985

/14/ RTE-A Utilities Manual, HP Manual Part Number 92077-90004, July 1985

/15/ RTE-A Relocatable Libraries Reference Manual, HP Manual Part Number 92077-90037, January 1985

ANHANG A

A.1 MD-ROUTINEN

Diese Routinen dienen dem Monitor zur Bearbeitung der Befehle und ihren Parametern.

A.1.1 MDSET

```
CALL MDSET (IDSET,IPTR,IPTRB)
```

IDSET	Integer*2	= 0	initialisiert die Eingaberoutine
		= 1	fordert einen neuen Eingabesatz
IPTR	Integer*2		Dummyparameter
IPTRB	Integer*2		Dummyparameter
IDSET		= 2	gibt den Eingabezeiger
		= 3	setzt den Eingabezeiger
IPTR			Wert des Eingabezeigers
IPTRB			Dummyparameter

A.1.2 MDTEXT

Liest einen Text mit folgendem Aufruf:

CALL MDTEXT (TEXT,NB,MAXBYT,IFLAG)

TEXT	Real*4,Dimension(*)	Feld, das einen Text in einem A1-Format beinhaltet
NB	Integer*2	Anzahl der gelesen Character
MAXBYT	Integer*2	Maximal erlaubte Zeichen für TEXT
IFLAG	Integer*2	= 0 alles in Ordnung < 1 Fehler in Eingabe > 1 zu viele Character wurden gelesen

A.1.3 MDUPLO

Wandelt Kleinbuchstaben in Großbuchstaben um.

CALL MDUPLO (TEXT,NB)

TEXT	Real*4,Dimension(*)	A1-Format, das umgewandelt werden soll
NB	Integer*2	Anzahl der Character, die umgewandelt werden sollen

A.1.4 MDPACK

Packt den Text in ein A1-Format mit bestimmter Länge.

```
ASK = MDPACK (TEXT,IPOS,LENGTH)
```

TEXT	Real*4,Dimension(*)	Werte, die umgepackt werden
IPOS	Integer*2	Startposition in TEXT
LENGTH	Integer*2	gibt das Am-Format zum Umpacken an (m zwischen 1 und 4)

A.1.5 MDA4CH

Kopiert ein A4-Format Array in einen Characterstring.

```
CALL MDA4CH (NB,ASK,CTEXT)
```

NB	Integer*2	Anzahl der Character in ASK
ASK	Real*4,Dimension(*)	Input array
CTEXT	Character	Ouputstring

A.1.6 MDEXP

Erweitert einen abgekürzten Commandstring und gibt die abgekürzte Folgenummer zurück.

```
CALL MDEXP (CTEXT,NB,CMND,LENSING,ICMND,IFOUND)
```

CTEXT	Character*8	Kommando, welches interpretiert wird
NB	Integer*2	Anzahl der Character, die gelesen werden
CMND	Character*8,Dim(*)	Vektor von Textfeldern, die abgefragt werden
LENSING	Integer*2,Dim(*)	Signifikante Länge der Befehle in CMND
ICMND	Integer*2	Anzahl der Befehle
IFOUND	Integer*2	= -1 CTEXT nicht in CMND vereinbart > 0 Position von CTEXT in CMND

A.1.7 MDFLT

Liest einen Real*4-Wert.

```
CALL MDFLT (VAL,SIG,IFLAG)
```

VAL	Real*4	gibt Real-Wert
SIG	Real*4	Vorzeichen von VAL
IFLAG	Integer*2	= 0 alles in Ordnung < 0 keine Zahl

A.2 COMPUTED GOTO

Um das Programm TAC in seine Unterprogramme zu verzweigen, wird ein Computed GOTO benutzt.

Der Aufruf dazu sieht fogendermaßen aus:

```
GOTO (s1,s2,s3,...) ,IFOUND
```

s1,s2,s3,... Marken von ausführenden Programmen

IFOUND Integer*2 Position von Text in CMND (aus MPEXP)

Wenn IFOUND kleiner oder gleich der Anzahl der ausführenden Anweisungen ist, wird die durch IFOUND bezeichnete Sprunganweisung ausgeführt.

Ist IFOUND größer als die Anzahl der ausführenden Anweisungen, wird die, der Sprunganweisung, folgende Anweisung ausgeführt.

Die Arbeit wurde in der Arbeitsgruppe für Schicht- und Ionentechnik (ASI) in der Kernforschungsanlage Jülich GmbH (KFA) durchgeführt.

Mein besonderer Dank gilt Herrn Dipl. Ing. U. Hacker für die Themenstellung und die Förderung der Arbeit durch wertvolle Hinweise, sowie das Durchsehen der Arbeit.

Dank sage ich auch den Mitarbeitern der Arbeitsgruppe für Schicht- und Ionentechnik, die durch ihre freundliche Hilfsbereitschaft mit zum Gelingen der Arbeit beigetragen haben.

Für die Betreuung seitens der Fachhochschule danke ich Herrn Prof. Dr. S. Pawelke und Herrn Prof. Dr. K. Schwarzer.

Diese Arbeit ist selbständig verfaßt und es sind keine anderen als die angegebenen Quellen und Hilfsmittel benutzt worden.

Jülich, im Februar 1986